Project Plan

Caffeine Coders | 25 June 2024

Fueling innovation, one cup at a time!

Document Table of Contents:

Deliverables List

We will produce 8 deliverables, as outlined in the provided "Group Project Assignment" document:

1. **Team Charter**
2. **Project Plan**
3. **Vision Document**
4. **Software Requirements Specification (SRS)**
5. **Skeletal System demo**
6. **Software Design**
7. **Minimal System demo**
8. **Target System demo**

Project Life Cycle

Our team will implement the **Staged delivery model** for the Trivial Compute game project. This approach is ideal because the requirements are well-defined, and our primary objective is to develop a highly reliable system that meets these specifications. The staged delivery model offers low overhead and enables all team members to contribute effectively throughout the development process.

The Trivial Computer Staged Delivery Lifecycle, as illustrated in the diagram below, outlines a structured approach to software development. It progresses through distinct stages to ensure comprehensive planning and execution:

1. **Planning and Definitions**: Establishing initial project goals and objectives.
2. **Requirements Analysis**: Identifying and documenting specific needs and criteria for the project.
3. **Architecture Design**: Creating the foundational blueprint for the system's structure.
4. **Incremental Development:** a. **Minimal Increment**: Delivering critical functionalities to ensure a working prototype. b. **Target Increment**: Building upon the foundation with more comprehensive features, aiming for completion within a semester. c. **Dream Increment**: Including aspirational features for development if additional time and resources become available.

Each increment involves a cycle of Detailed Design, Coding, Testing, and Delivery.

This staged delivery model ensures a methodical and scalable approach to software development, facilitating continuous improvement and adaptability. It allows us to

prioritize features, manage resources effectively, and deliver a functional product at each stage while working towards the complete vision of the Trivial Compute game.

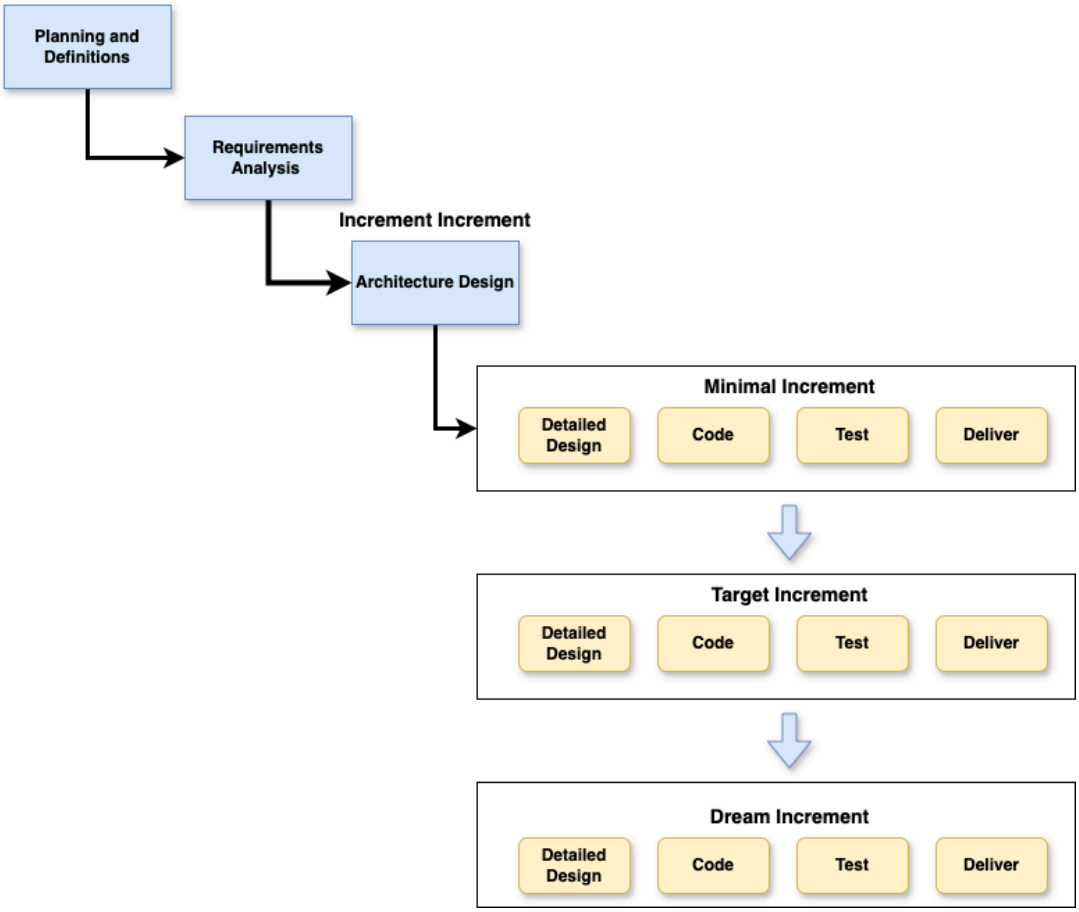


Figure 1: Trivial Computer Staged Delivery Lifecycle

Increment Features

Skeletal Increment

The skeletal increment architecture, as depicted in the diagram, presents an integrated framework for a robust and scalable web application. The architecture is composed of several key components:

1. **Core Server:** An Express server, built on Node.js, forms the backbone of the application. This server manages hosting responsibilities and handles the application's core logic.

2. **Database and Authentication:** The server integrates Firebase SDK, connecting to Google's Realtime Database for efficient data management and leveraging Firebase Authentication for secure user verification.
3. **API Endpoints:** REST endpoints are implemented to facilitate seamless communication and data exchange between the client and server.
4. **Containerization:** The entire web application is encapsulated within a Docker container, ensuring consistency across different environments and simplifying deployment processes.
5. **User Interface:** The client-side interface is developed using HTML5, providing a dynamic and responsive user experience. This interface interacts with the server to render data and handle user inputs.
6. **Client-Server Communication:** The architecture enables efficient handling of requests and responses between the client interface and the server, maintaining a reliable and interactive user experience.

This skeletal architecture provides a solid foundation for building a feature-rich web application. It offers scalability, allowing for future expansions and modifications, while maintaining performance and security. The use of modern technologies like Node.js, Express, Firebase, and Docker ensures that the application can leverage current best practices in web development and deployment.

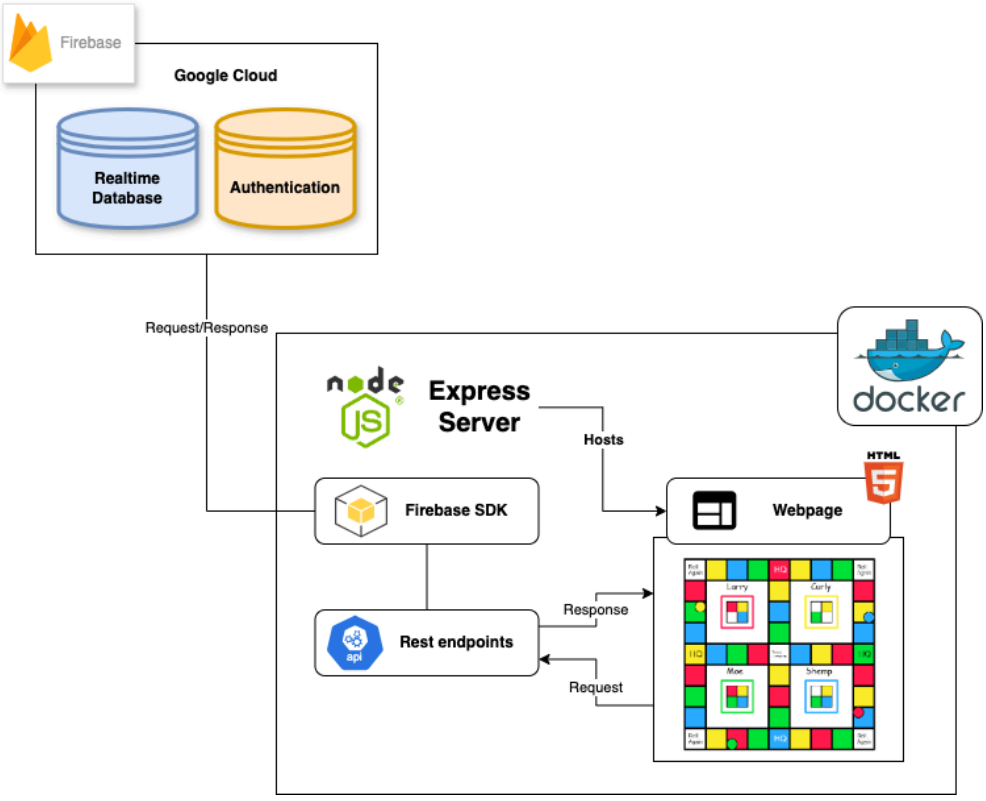


Figure 2: Skeletal Architecture Trivial Compute

Minimal Increment

In the Minimal increment, we will deliver the core functionality of our project, focusing on essential game operations through a robust backend and responsive frontend. Our primary goal is to ensure that the fundamental elements of the game are operational, providing a solid foundation for future enhancements.

Key Features:

1. **Game Board Web GUI:** An intuitive and user-friendly visual interface where players interact with the game.
2. **Player Movement Logic:** Handles player movement across the game board, ensuring smooth and accurate interactions within the game environment.
3. **Score Keeping and End State Logic:** Tracks player scores and determines game session endings, providing accurate feedback and results to players.
4. **Basic User Interface:** Facilitates gameplay, making the game accessible and enjoyable for users.

Technical Stack:

1. **Backend:** Node.js with Express.js framework, offering a reliable and scalable server-side solution.
2. **Frontend:** JavaScript, HTML, and CSS for a responsive and interactive user experience.
3. **Database:** Firebase Realtime Database for efficient and secure data storage and retrieval.
4. **System Communication:** REST API for smooth and efficient data exchange between client and server components.

This Minimal increment establishes the core gameplay functionality and essential user interactions. It provides a functional game experience while laying the groundwork for additional features and refinements in subsequent increments. The chosen technical stack ensures scalability, performance, and ease of future development, aligning with modern web development practices.

Target Increment

The Target increment builds upon the Minimal increment, encompassing all features and functionalities we aim to develop over one semester. This phase significantly enhances the initial version of the project, delivering a more comprehensive, secure, and user-friendly game experience.

Key Enhancements:

1. **User Authentication:**
    - Implement a robust login system
    - Ensure secure access for all players
    - Integrate with Firebase Authentication for reliable user management
2. **Dynamic Content Management:**
    - Add functionality to create and delete trivia cards
    - Enable a customizable and evolving game experience
    - Implement database operations for efficient content handling
3. **Enhanced User Interface:**
    - Develop advanced game settings features
    - Provide players with more control over gameplay options
    - Improve overall user experience and interface design
4. **Secure Deployment:**
    - Utilize Docker for containerization
    - Ensure a consistent environment across development and production stages
    - Implement best practices for secure and scalable deployment
5. **Advanced Game Logic:**
    - Refine score-keeping mechanisms
    - Implement more complex game rules and scenarios
    - Enhance end-game states and player progression
6. **Performance Optimization:**
    - Improve load times and responsiveness
    - Optimize database queries and API calls
    - Enhance overall system efficiency

This Target increment represents a significant step forward in the project's development, offering a feature-rich, secure, and highly customizable gaming platform. By focusing on user authentication, content management, and advanced gameplay features, we aim to deliver a polished product that meets the semester's objectives while laying the groundwork for potential future enhancements.

Dream Increment

The Dream increment represents an aspirational vision for our project, outlining potential features that could be developed given additional time and resources. This phase aims to significantly enhance the user experience and game complexity through advanced technologies and innovative gameplay elements.

Key Aspirational Features:

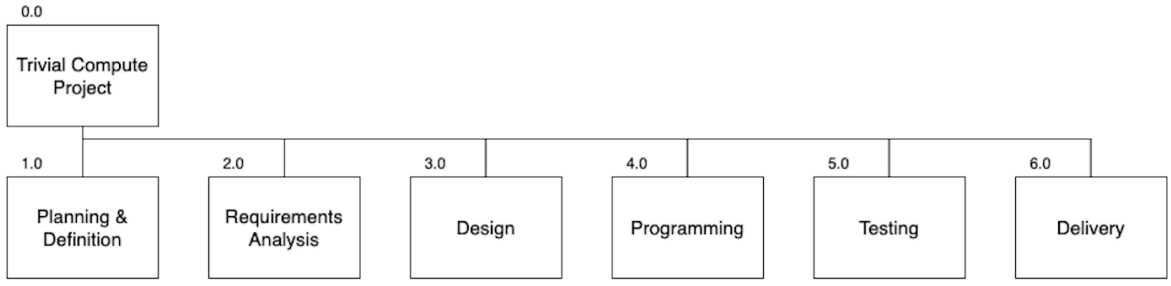
1. **AI-Powered Non-Playable Characters (NPCs):**
    - Integrate large language models, such as those provided by the OpenAI API
    - Enable users to play against computer-controlled opponents
    - Simulate intelligent and dynamic NPC behavior
2. **Customizable NPC Difficulty Levels:**
    - Easy: NPC provides correct answers 50% of the time
    - Medium: NPC provides correct answers 70% of the time
    - Hard: NPC provides correct answers 90% of the time
3. **Enhanced Game Modes:**
    - Implement single-player campaigns against NPCs
    - Create multiplayer modes combining human players and NPCs

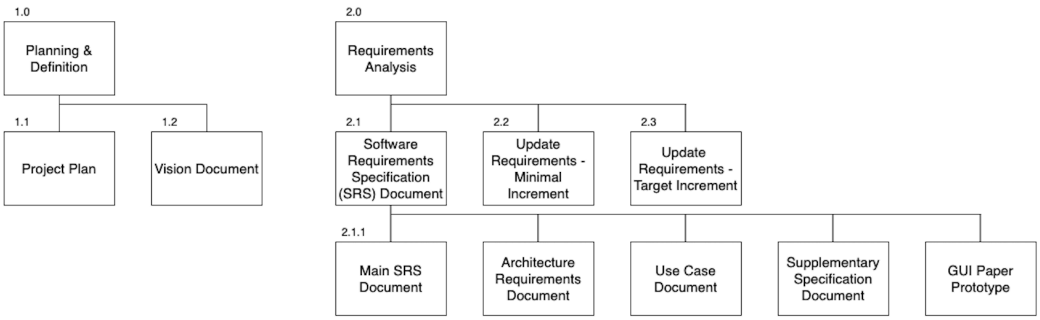
These Dream increment features would dramatically expand the game's depth and replayability. We could offer a more diverse and challenging gameplay experience by incorporating AI-powered NPCs with adjustable difficulty levels.

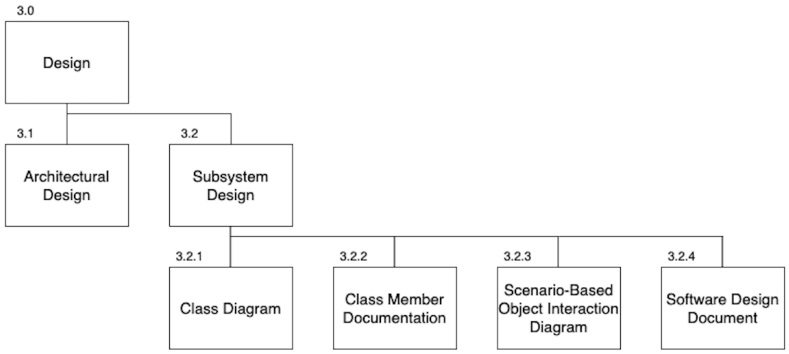
While these features are beyond the scope of our current development timeline, they represent exciting possibilities for future expansion and innovation in our project.

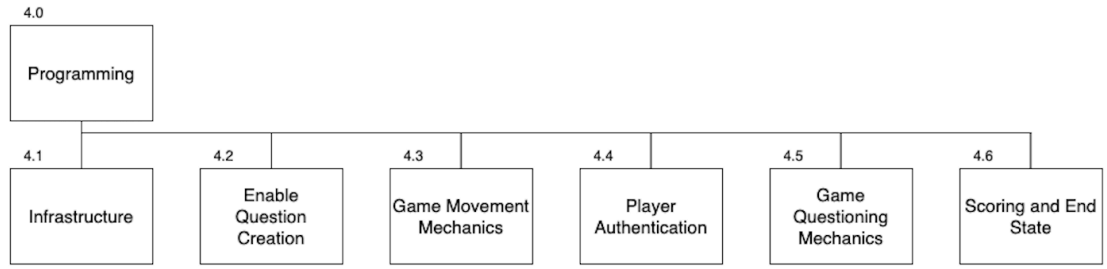
Work-Breakdown Structure (WBS)

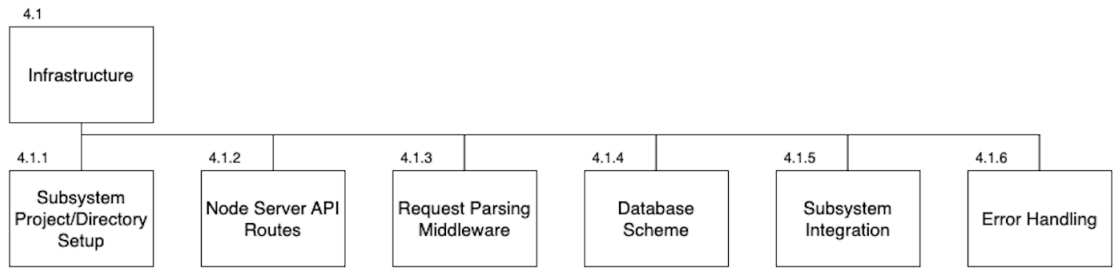
Below is Caffeine Coder’s Work Breakdown Structure:

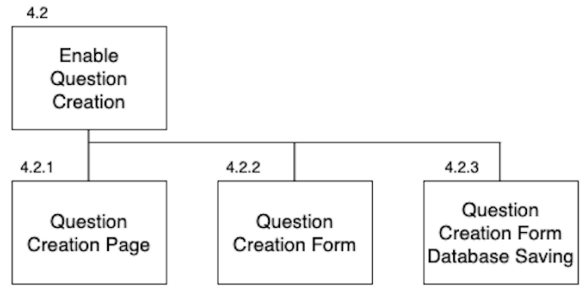


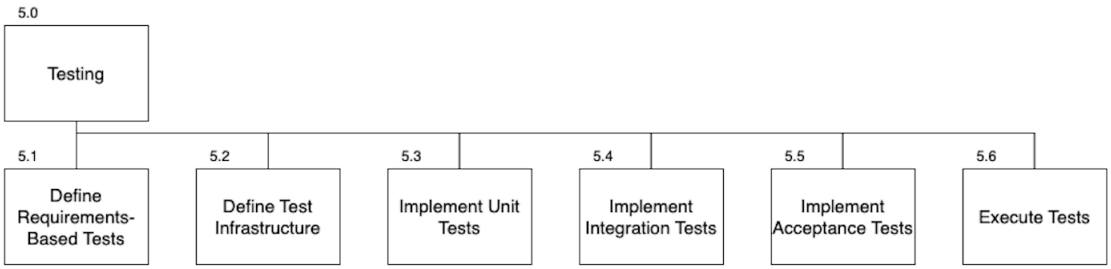

1.0
Planning & Definition
1.1
Project Plan
1.2
Vision Document
2.0
Requirements Analysis
2.1
Software Requirements Specification (SRS) Document
2.2
Update Requirements - Minimal Increment
2.3
Update Requirements - Target Increment
2.1.1
Main SRS Document
Architecture Requirements Document
Use Case Document
Supplementary Specification Document
GUI Paper Prototype

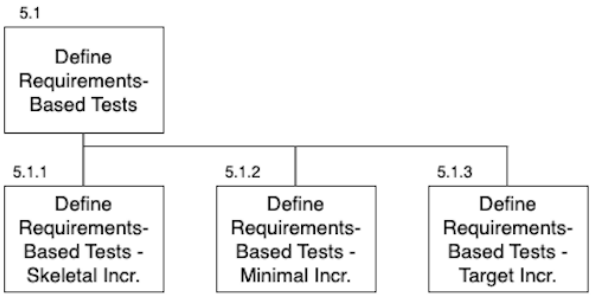

3.0
Design
3.1
Architectural Design
3.2
Subsystem Design
3.2.1
Class Diagram
3.2.2
Class Member Documentation
3.2.3
Scenario-Based Object Interaction Diagram
3.2.4
Software Design Document

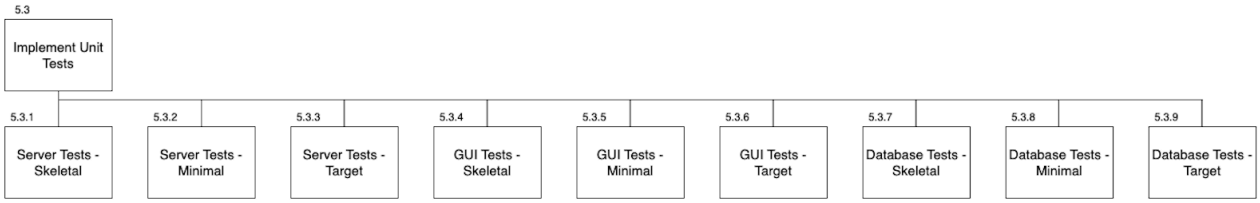

4.0
Programming
4.1
Infrastructure
4.2
Enable Question Creation
4.3
Game Movement Mechanics
4.4
Player Authentication
4.5
Game Questioning Mechanics
4.6
Scoring and End State

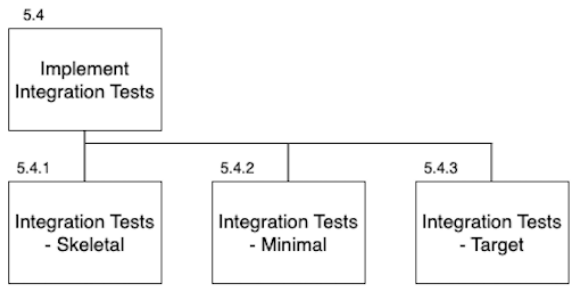

4.1
Infrastructure
4.1.1
Subsystem Project/Directory Setup
4.1.2
Node Server API Routes
4.1.3
Request Parsing Middleware
4.1.4
Database Scheme
4.1.5
Subsystem Integration
4.1.6
Error Handling

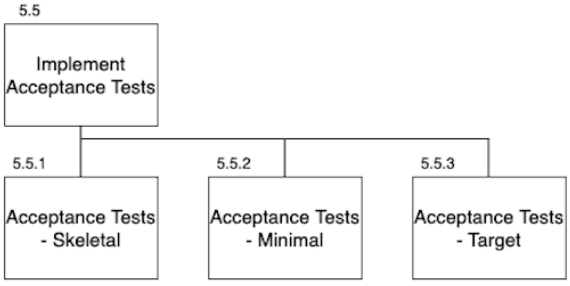

4.2
Enable Question Creation
4.2.1
Question Creation Page
4.2.2
Question Creation Form
4.2.3
Question Creation Form Database Saving

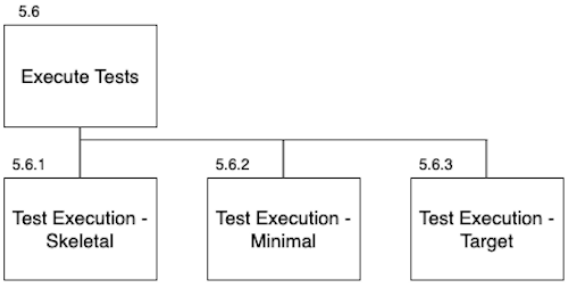

5.0
Testing
5.1
Define Requirements-Based Tests
5.2
Define Test Infrastructure
5.3
Implement Unit Tests
5.4
Implement Integration Tests
5.5
Implement Acceptance Tests
5.6
Execute Tests

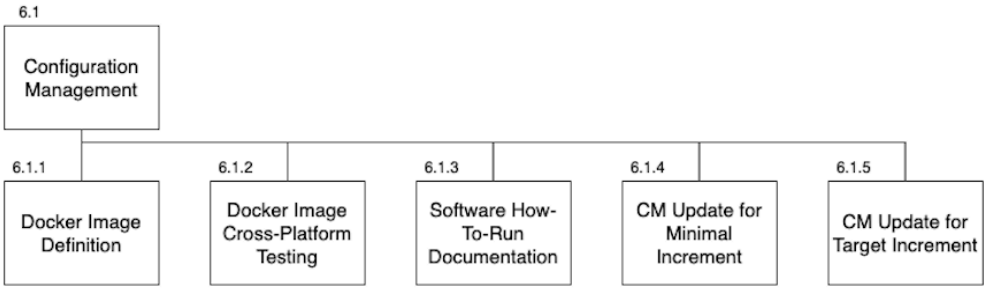

5.1
Define Requirements-Based Tests
5.1.1
Define Requirements-Based Tests - Skeletal Incr.
5.1.2
Define Requirements-Based Tests - Minimal Incr.
5.1.3
Define Requirements-Based Tests - Target Incr.


5.3
Implement Unit Tests
5.3.1
Server Tests - Skeletal
5.3.2
Server Tests - Minimal
5.3.3
Server Tests - Target
5.3.4
GUI Tests - Skeletal
5.3.5
GUI Tests - Minimal
5.3.6
GUI Tests - Target
5.3.7
Database Tests - Skeletal
5.3.8
Database Tests - Minimal
5.3.9
Database Tests - Target


5.4
Implement Integration Tests
5.4.1
Integration Tests - Skeletal
5.4.2
Integration Tests - Minimal
5.4.3
Integration Tests - Target


5.5
Implement Acceptance Tests
5.5.1
Acceptance Tests - Skeletal
5.5.2
Acceptance Tests - Minimal
5.5.3
Acceptance Tests - Target


5.6
Execute Tests
5.6.1
Test Execution - Skeletal
5.6.2
Test Execution - Minimal
5.6.3
Test Execution - Target



Quality Plan

Project Tracking & Control

To ensure effective monitoring of project progress and adherence to the project plan schedule, we will implement a systematic approach as outlined in the following flow chart:

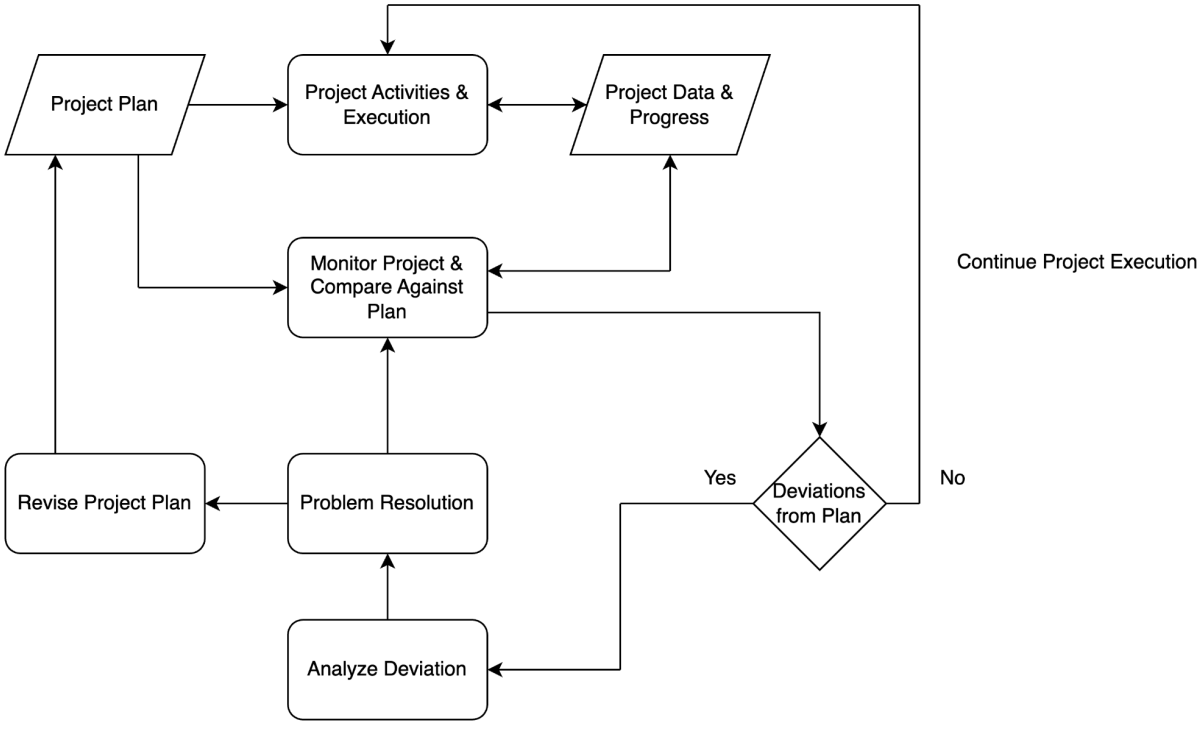


Figure 3: Project tracking and control process

Our project tracking and control process involves the following key steps:

1. **Project Plan Baseline:**
    - Establish a comprehensive project plan with detailed schedules for all tasks.
    - Set clear milestones and deliverables for each project phase.
2. **Continuous Monitoring:**
    - Regularly monitor artifacts, deliverables, and project data.
    - Compare actual progress against the baseline project plan.
3. **Variance Analysis:**
    - Identify any deviations from the planned schedule or deliverables.
    - Analyze the causes and potential impacts of these deviations.
4. **Corrective Action:**
    - For identified deviations, develop and implement corrective actions.
    - Adjust resources, timelines, or scope as necessary to address variances.
5. **Project Plan Updates:**
    - Update the project plan to reflect any changes resulting from corrective actions.
    - Ensure all team members are informed of and aligned with these updates.
6. **Continued Execution:**
    - In the absence of deviations, continue project execution as planned.
    - Maintain vigilance in monitoring to quickly identify any future variances.
7. **Iterative Process:**
    - Repeat this cycle throughout the project lifecycle.
    - Use insights gained to continuously improve project management processes.

This iterative approach enables our team to effectively track schedule variance, manage risks, and ensure that the project remains on course to meet its objectives. By maintaining a proactive stance in project tracking and control, we can swiftly address challenges and capitalize on opportunities, ultimately leading to successful project delivery.

Quality Assurance & Requirements Management

Requirements Development

1. **Stakeholder Analysis**: Perform a thorough analysis of project stakeholders to generate detailed requirements. This involves identifying all stakeholders, their needs, and how to effectively address them.
2. **Functional/Non-Functional Requirements**: Develop use cases to highlight functional requirements. Each use case will demonstrate a specific scenario and detail the user and system's responses. Non-functional requirements will be identified using a checklist in the supplementary specification document.
3. **Peer Reviews**: Conduct peer reviews of the outlined requirements to ensure they are clear, correct, unambiguous, and testable. This aims to identify inconsistencies early in the process.

Documentation

1. **Initial Documentation**: Document the initial set of requirements in the vision document and SRS document increments. This will serve as the baseline for the project.
2. **Incremental Updates**: Update the requirement documents at each increment of our staged deliveries. This allows for flexibility and accommodates any changes or new requirements that may emerge during the project lifecycle.

Quality Objectives

1. **Objective Translation**: Translate the requirements into clear, measurable quality objectives for each deliverable. These objectives will be specific, achievable, and relevant.
2. **Acceptance Test Basis**: Use these quality objectives as the basis for developing our project's acceptance tests. This ensures that deliverables meet the defined quality criteria and stakeholder expectations.

Iterative Improvements

1. **Feedback Loop**: Analyze project data to continuously improve the quality assurance process. This step enables the team to identify lessons learned and implement this feedback for subsequent increments. Update requirement documents as necessary.

Testing

This section outlines our comprehensive approach to testing, ensuring thorough project coverage. Performing these tests is critical for validating functionality, tracking progress, and meeting stakeholder expectations.

Test Strategy

Our testing strategy incorporates various levels of software testing to ensure quality and reliability:

1. **Unit Testing**: Verify functionality of individual components.
2. **Integration Testing**: Verify interactions between integrated subsystems.
3. **Acceptance Testing**: Validate that the software meets the quality criteria defined in the requirements to meet stakeholder expectations.

Test Outline

The tests will be implemented and documented using a series of parameters:

1. **Objectives**: Define the expected outcomes.
2. **Scope**: Define the features, subsystems, or interfaces to be tested.
3. **Methodologies**: Define the tools, techniques, and environments for testing.

Test Execution

Tests will be designed and executed as follows:

1. **Test Cases**: Design test cases that clearly define inputs and expected outputs.
2. **Test Execution**: Perform the tests and document all results.

Result Analysis

The data produced from the tests will be analyzed and used to update the project:

1. **Corrective Action**: Perform rework to improve metrics as necessary. Update the project plan and requirement documents.
2. **Report**: Summarize and document the generated data. Use this information to update the project plan.

Configuration Management

Considering that this project is expected to produce numerous artifacts, including documents and demos, configuration management will be utilized to effectively control changes to configuration items. The following processes will be implemented to preserve the coherence of project deliverables:

Configuration Identification

1. **Artifact Identification**: Identify all the project's expected artifacts to be considered for control.
2. **Peer Review**: Conduct a peer review of the selection and document a master list.
3. **Unique Identifiers**: Create unique identifiers for the artifacts.

Establish Baselines

1. **Baseline Creation**: Establish baseline artifacts once they are reviewed and approved.
2. **Documentation**: Document baselines with dates and versions.

Configuration Control

1. **Change Requests**: Document and submit change requests when configuration changes are needed.
2. **Change Evaluation**: Evaluate proposed changes concerning the project plan.
3. **Approval and Update**: The Configuration Management (CM) Engineer will approve and update changes to the baseline.
4. **Record Maintenance**: Maintain records of all configuration items and their status.

Version Control

1. **Software Version Control**: Utilize GitHub to maintain version control of the software.
2. **Document Tracking**: Track document changes using unique identifiers and versions.
3. **Merge Evaluation**: The CM Engineer will evaluate merges.

Configuration Audits and Reviews

1. **Audits**: Perform audits to verify adherence to the configuration management procedures.
2. **Reviews**: Review configuration baselines and updates to ensure compliance.

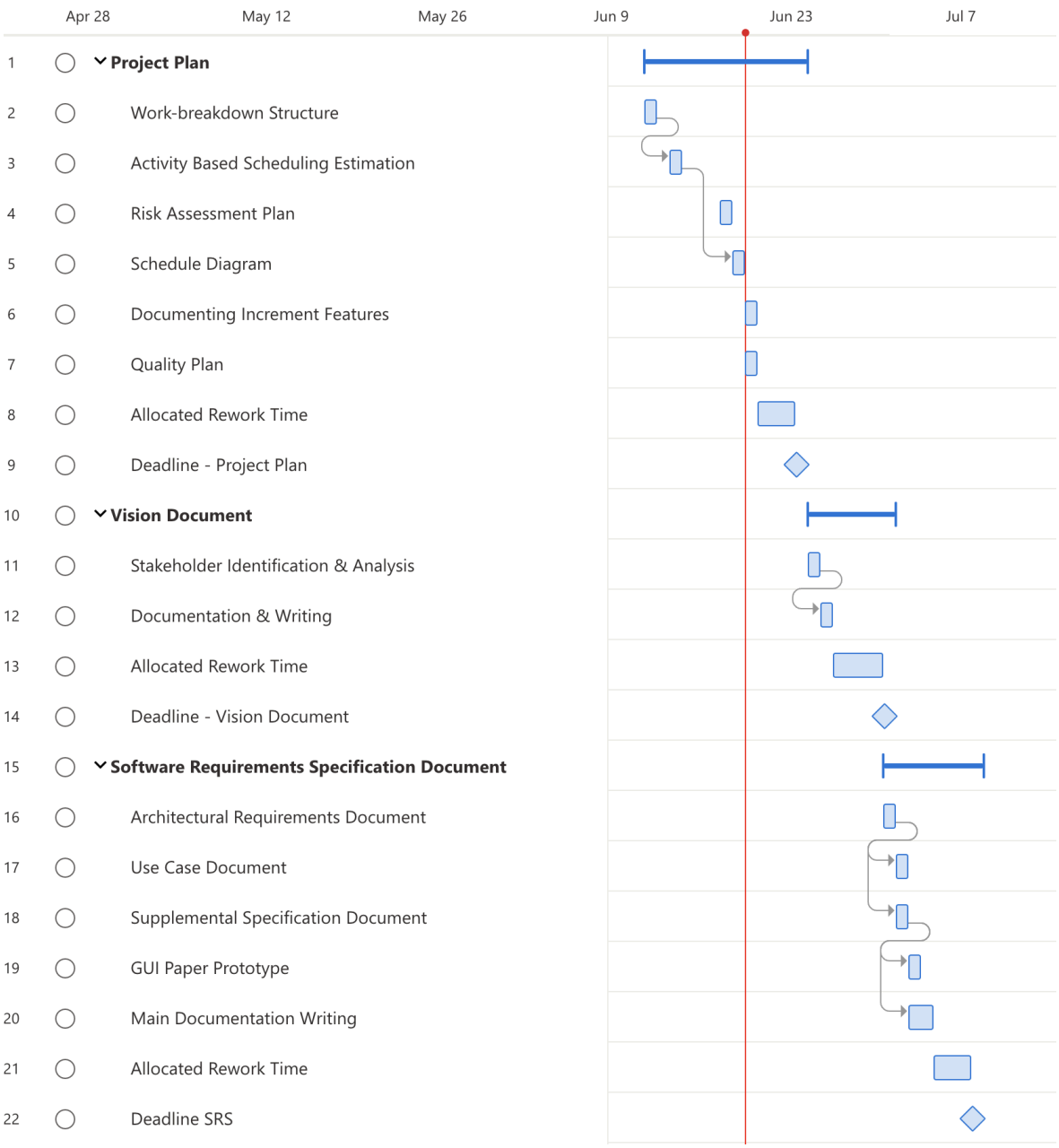
Project Estimates & Resourcing

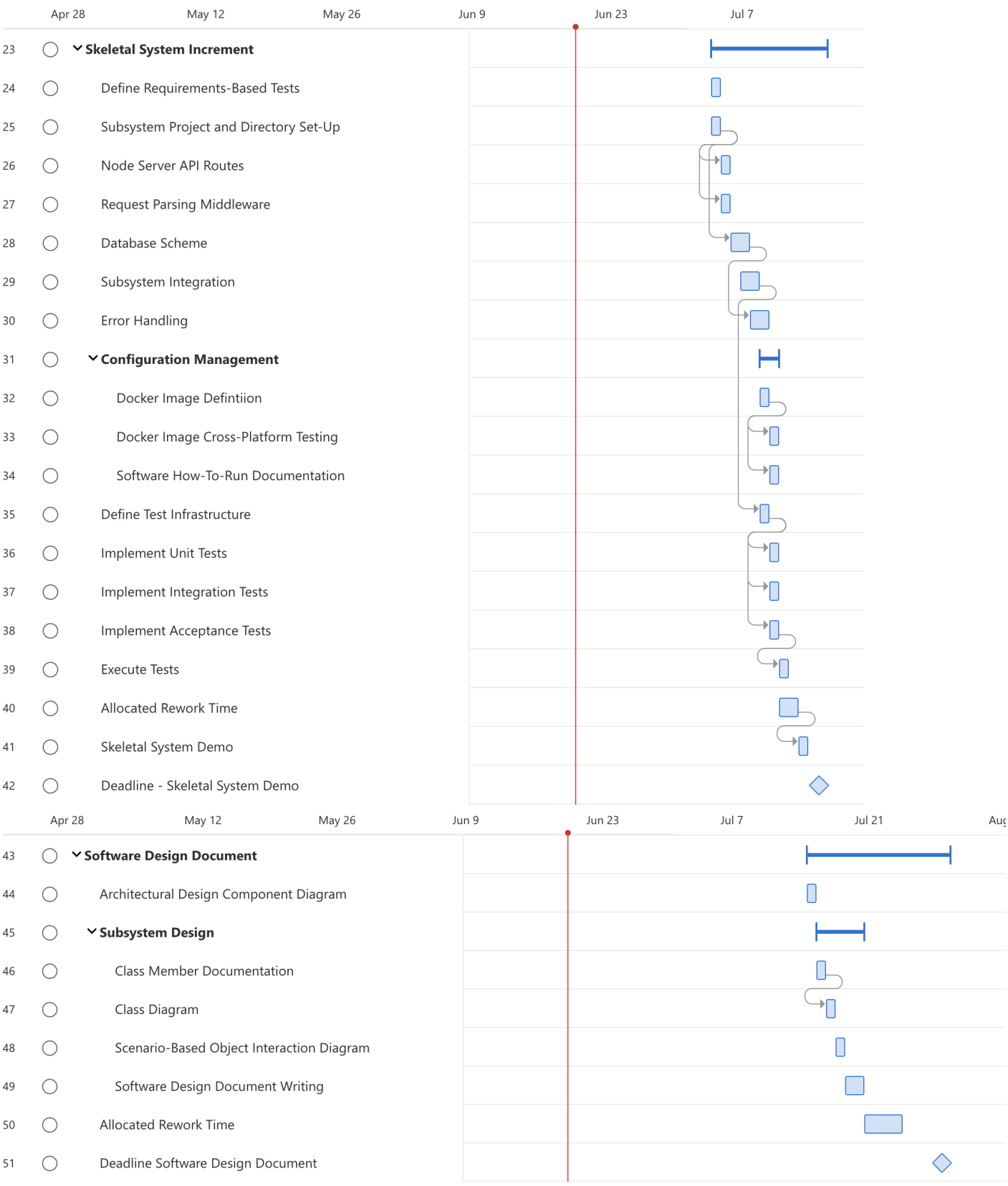
Our team used the two round of Delphi estimation method to generate detailed estimates for each task in the project. The estimates are broken down into a hierarchical structure, covering major project phases and their sub-tasks. Here's an overview of our estimation process and results:

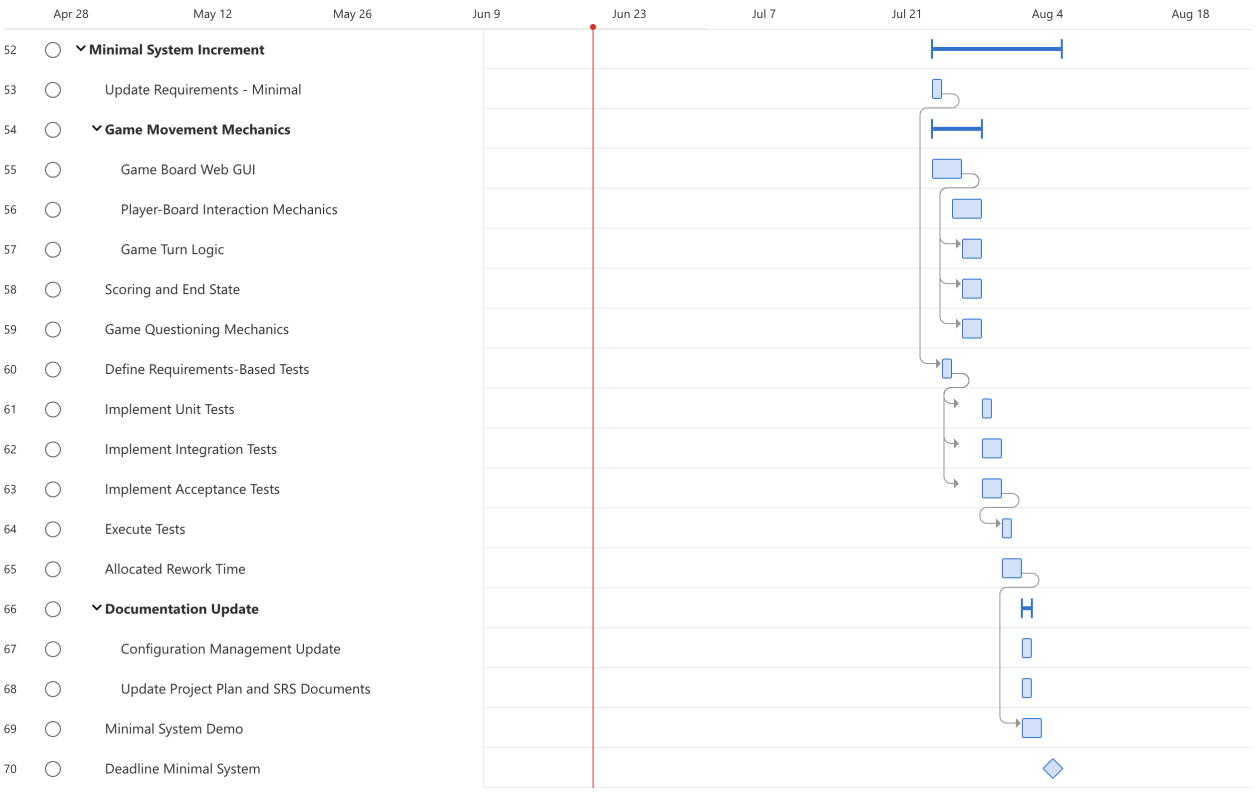
1. **Estimation Method:**
    - We used a three-point estimation technique (optimistic, pessimistic, and most likely)
    - The final estimate is calculated as an average: (optimistic + 4 * most likely + pessimistic) / 6
2. **Time Units:**
    - Estimates are provided in person-hours
    - Calendar days are calculated assuming 4 person-hours per day
3. **Project Structure:**
    - The project is divided into main phases: Planning & Definition, Requirements Analysis, Design, Programming, Testing, and Delivery
    - Each phase is further broken down into specific tasks and sub-tasks
4. **Key Estimates:**
    - Total project effort: Approximately 229 person-hours
    - Total calendar days: 58 days (rounded up from 57.3)
5. **Major Phase Estimates (approximate person-hours):**
    - Planning & Definition: 12.5
    - Requirements Analysis: 19.25
    - Design: 21
    - Programming: 55.5
    - Testing: 67.5
    - Delivery: 32
6. **Resource Allocation:**
    - The estimates assume a team of 4 members working part-time on the project
    - Daily work capacity is set at 4 person-hours per day across the team
7. **Uncertainty Handling:**
    - Each task includes optimistic and pessimistic estimates to account for uncertainty
    - The three-point estimation method helps to balance overly optimistic or pessimistic views

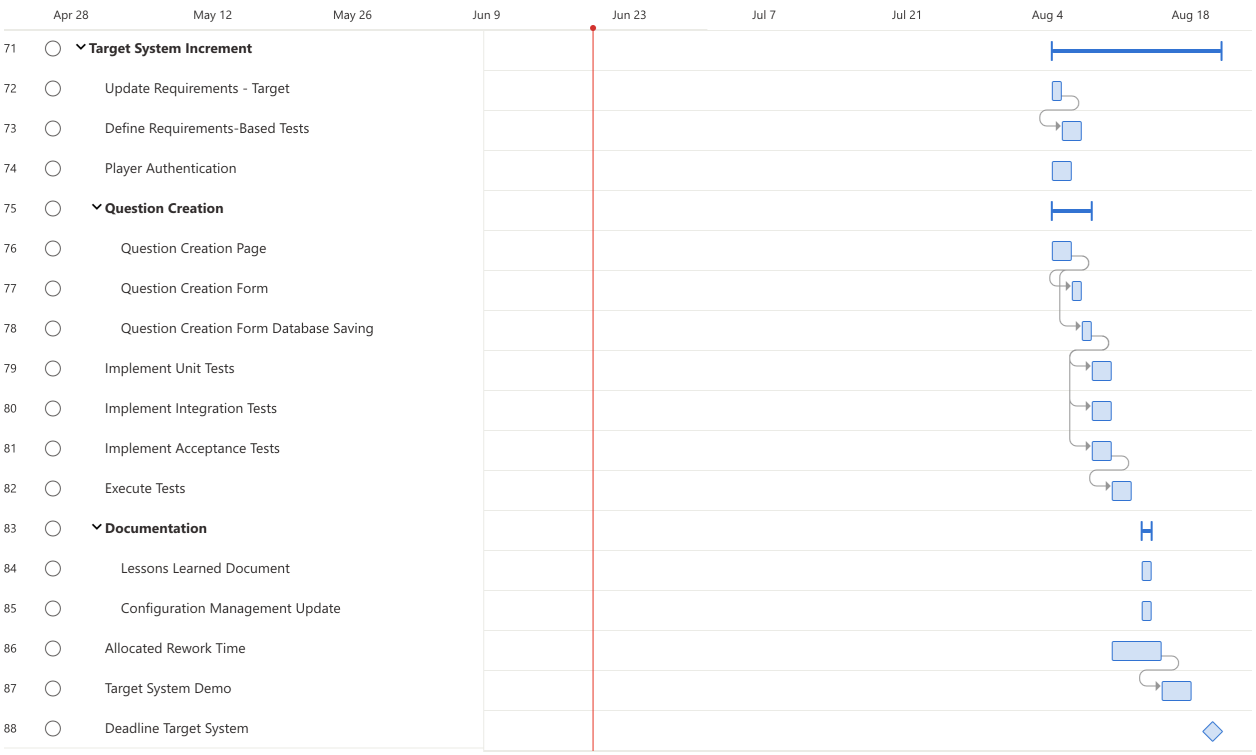
Schedule & Dependencies

This section outlines the timeline for our project, detailing the start and end dates for each deliverable. A Gantt chart is used to visualize the project schedule and highlight the task dependencies, providing a clear overview of the project's progress.




Apr 28
May 12
May 26
Jun 9
Jun 23
Jul 7
Jul 21
23 Skeletal System Increment
24 Define Requirements-Based Tests
25 Subsystem Project and Directory Set-Up
26 Node Server API Routes
27 Request Parsing Middleware
28 Database Scheme
29 Subsystem Integration
30 Error Handling
31 Configuration Management
32 Docker Image Defintiion
33 Docker Image Cross-Platform Testing
34 Software How-To-Run Documentation
35 Define Test Infrastructure
36 Implement Unit Tests
37 Implement Integration Tests
38 Implement Acceptance Tests
39 Execute Tests
40 Allocated Rework Time
41 Skeletal System Demo
42 Deadline - Skeletal System Demo
43 Software Design Document
44 Architectural Design Component Diagram
45 Subsystem Design
46 Class Member Documentation
47 Class Diagram
48 Scenario-Based Object Interaction Diagram
49 Software Design Document Writing
50 Allocated Rework Time
51 Deadline Software Design Document





The above Gantt chart was produced using Microsoft Project 3. The custom workweek was set to Monday - Sunday with each day consisting of 4 person-hours. Task durations

were estimated using the Delphi method as mentioned above in the Project Estimates section.

Risk Assessment & Risk Plan

Developing the Trivial Computer game presents several risks for Caffeine Coders. Here, we outline the identified risks along with strategies to mitigate them.

Risk Identification

Below are the identified Risk The Caffeine coders may face:

#	Risk Category	Risks
1	Concerns	IF members of their team do not have the time to work on the class due to their day jobs, THEN some of the deliverables may not be met.
2	Issues	IF the target system does not accept containers THEN our server will not be deployable.
3	Uncertainty	IF the user interface is too cumbersome, THEN the users may not continue to play the game.
4	Issues	IF third party dependencies and services (firebase) reach their free tier limit THEN we will exceed budget for the course.
5	Knowledge	IF the database access rules are not set THEN we may have a break in confidentiality of the data.
6	Uncertainty	IF the chosen third party services unexpectedly change or discontinue their services, THEN our project may not function properly.
7	Uncertainty	IF the game UI and instructions are not intuitive and clear, THEN users may find it unplayable
8	Concerns	IF a team member unexpectedly leaves the project, THEN their knowledge may be lost

Risk Analysis

The chart below represents Caffeine Coders Risk Matrix. Each risk is numbers 1-8 using the numbers identified above

			Potential Consequence				
			L6	L5	L4	L2	L1
			Minor impacts to schedule	Impacts to schedule that lead to increased time spent by the members on the team	Impacts to schedule the lead to late delivery	Missing of Operational Capabilities	Failure to Deliver Capabilities
Likelihood	>80%	Highly Likely	High	Very High	Very High	Very High	Very High
	61-80%	Likely	Medium	High	Very Hig 3	Very Hig 7	Very Hi 1
	41-60%	Improbable	Low	Medium	High	Very High	Very High
	12-40%	Unlikely	Low	Low	Medium	H 5 2	High
	1-20%	High Likely	Low	Low	6 Low	4 edium	Mediu 8

Risk Prioritization based on the risk matrix

Risk Priority	Risk Number
1	1
2	7
3	3
4	2
5	5
6	8
7	4
8	6

Risk Planning & Resolution

#	Mitigations
1	The team lead will assign deadlines for each task. Communications channels like discord will be used to help keep the team on the same page with deliverables.
2	We will work with the end user to identify deployment standards that include container runtime environments.
3	Conduct delivery in increments to get user feedback and iterate.
4	Identify alternative databases like mongodb or postgres.
5	Develop unit tests for accessing the database.
6	Leverage stable versions of dependencies.
7	Create necessary error handling and instructions to help the users
8	Communicate roles and responsibilities with the team so that if someone leaves, we can track what they were working on.

Risk Matrix After Mitigations are applied.

			Potential Consequence				
			L6	L5	L4	L2	L1
			Minor impacts to schedule	Impacts to schedule that lead to increased time spent by the members on the team	Impacts to schedule the lead to late delivery	Missing of Operational Capabilities	Failure to Deliver Capabilities
Likelihood	>80%	Highly Likely	High	Very High	Very High	Very High	Very High
	61-80%	Likely	Medium	High	Very Hig 3	Very Hig 7	Very Hi 1
	41-60%	Improbable	Low	Medium	High	Very High	Very High
	12-40%	Unlikely	Low	Low	Medium	5 2	High
	1-20%	High Likely	Low	Low	6 Low	4 dium	M 8 m

Risk Tracking

Risk tracking is a critical component of Caffeine Coders' project management strategy. Our team will implement a comprehensive and proactive risk management approach throughout the project lifecycle:

1. **Risk Identification:**
    - Initial risks are identified during the project planning phase.
    - Team members are encouraged to report new risks as they arise during the project.
2. **Risk Documentation:**
    - All identified risks are documented in a centralized risk register.
    - Risks are visualized and prioritized using risk matrices.
3. **Regular Risk Assessment:**
    - Bi-weekly risk assessment meetings are scheduled to review and update the risk register.
    - The team evaluates the status of existing risks and identifies any new potential threats.
4. **Risk Communication:**
    - Updates to the risk register are communicated to all team members promptly.
    - A summary of key risks is included in project status reports to stakeholders.
5. **Mitigation Strategies:**
    - For each significant risk, the team develops and documents specific mitigation strategies.
    - Responsibilities for implementing mitigation actions are clearly assigned.
6. **Contingency Planning:**
    - Contingency plans are developed for high-priority risks to ensure quick response if they materialize.
    - These plans are reviewed and updated regularly to ensure their relevance.
7. **Risk Monitoring:**
    - Key risk indicators are established and monitored throughout the project.
    - The effectiveness of implemented mitigation strategies is evaluated regularly.
8. **Continuous Improvement:**
    - Lessons learned from risk management are documented and used to improve future risk tracking processes.

By maintaining this structured and proactive approach to risk management, Caffeine Coders aims to minimize uncertainties, enhance decision-making, and increase the

likelihood of successful project delivery. This process ensures that our team remains vigilant and adaptable in the face of potential challenges throughout the software development lifecycle.

Work Responsibility Matrix

Casey Rock

- Lead Architect
    - Will be responsible for defining the high-level approaches for each of the server, GUI, and database.
    - Defines software interfaces among all 3 components.
    - May write diagrams and/or code outlines (blank classes/functions with docstrings for example) to express high-level interfaces.
- Lead CM Engineer
    - Will be responsible for ensuring users can run our software regardless of their environment.
    - Produces technical documentation for how to use the software and any setup needed.
    - Defines what the software dependencies for the project are and where they should be sourced from.
- Programmer

Calvin Nguyen

Roles/Responsibilities:

- Lead Tester
    - Will be responsible for ensuring the project meets the assignment requirements on the implementation side.
    - Implements and runs tests based on requirements defined by the SQA engineer.
    - Works with programmers to ensure all tests pass.
- Programmer
    - Will write code as needed for server, GUI, and/or database.

Justin Cheung

Roles/Responsibilities:

- Project Manager
    - Will be responsible for assigning tasks and ensuring our team meets deadlines on all deliverables.

    - Performs final minor edits on deliverables and submits them by the appropriate deadlines.
- Lead SQA Engineer
    - Will be responsible for ensuring the project meets the assignment requirements on the non-implementation side.
    - Collects & communicates requirements for each deliverable.
    - Clarifies requirements with users (professors) as needed.
    - Defines unambiguously what tests need to be created to cover the requirements.
- Programmer
    - Will write code as needed for server, GUI, and/or database.

Ting Wei Wang

Roles/Responsibilities:

- Lead Programmer
    - Guide the development team's technical strategy and ensure high code quality through reviews and best practices.
    - Collaborate with the lead architect to create detailed plans for the implementation of system architecture to ensure scalability, stability, and performance.
    - Track progress for timely software delivery.
- Lead Presenter
    - Organize and prepare clear, concise, and engaging presentations for stakeholders and team members.
    - Translate complex technical concepts into easily understandable terms for non-technical audiences.
    - Build and maintain strong relationships with stakeholders through regular, professional communication and feedback incorporation.

Assumptions & Constraints

Assumptions:

1. **Team Availability:** All team members will be available for the duration of the project, dedicating the agreed-upon hours per week.
2. **Technical Skills:** Team members possess the necessary skills to work with the chosen technologies (Node.js, Express, Firebase, Docker, etc.).
3. **Resource Access:** The team will have access to all required development tools and platforms throughout the project lifecycle.

4. **Stakeholder Engagement:** Stakeholders (professors) will be available for timely feedback and clarification of requirements as needed.

Constraints:

1. **Time:** The project must be completed within one semester, with specific deadlines for each deliverable as outlined in the course schedule.
2. **Budget**: The project must be completed using free-tier services and tools to avoid exceeding the course budget.
3. **Technology Stack:** The project must use the technologies specified in the project plan (Node.js, Express, Firebase, Docker, etc.).
4. **Deliverables:** The project must produce the 8 specific deliverables outlined in the Deliverables List section of the project plan.
5. **Academic Integrity:** All work must adhere to the university's academic integrity policies and guidelines.
6. **Team Size:** The project must be completed by the current team of four members, with no additional resources.

Lesson Learned

1. **Planning Process:**
    - Collaborative planning using tools like shared documents and version control systems improved team coordination and document quality.
    - The Delphi method for estimations provided more accurate and well-rounded estimates than individual assessments.
2. **Time Management:**
    - Early establishment of a detailed project timeline helped in visualizing the project scope and identifying potential bottlenecks.
    - Allocating buffer time for unforeseen circumstances proved crucial in maintaining schedule flexibility.
3. **Team Communication:**
    - Regular team meetings and use of communication platforms like Discord enhanced information sharing and problem-solving.
    - Clear definition of roles and responsibilities from the outset minimized confusion and improved task allocation.
4. **Technology Stack:**
    - Researching and agreeing on the technology stack early in the planning phase allowed for better resource allocation and skill alignment.
5. **Risk Management:**
    - Proactive identification and analysis of risks during the planning phase prepared the team for potential challenges.
    - The risk matrix proved to be an effective tool for prioritizing and visualizing project risks.

6. **Requirement Gathering:**
    - Thorough stakeholder analysis helped in capturing comprehensive project requirements.
    - Iterative requirement refinement through peer reviews improved the clarity and testability of requirements.
7. **Continuous Improvement:**
    - The importance of maintaining flexibility in the project plan to accommodate new insights and changing circumstances was recognized.
8. **Documentation**:
    - Detailed documentation of project processes and decisions in the planning phase is expected to serve as a valuable reference throughout the project lifecycle.

These lessons learned reflect insights gained during the project planning phase and set the stage for continuous improvement throughout the project execution. As the project progresses, the team should continue to document new lessons learned to inform future projects and enhance overall project management skills.

Credits List

Justin planned the timeline for project plan document delivery and wrote the WBS.

Casey defined the architecture, wrote the risk plan, and wrote the initial programming task list.

Calvin defined project task dependencies and created the Gantt chart schedule.
Ting-Wei wrote the initial increment breakdown, review and revise the final document.
All authors contributed to significant edits across the document and discussion/feedback regarding the contents of the document.

Authors worked collaboratively on the Increment Features section. All authors were involved with each Delphi estimation iteration.